REMARKS

In view of the following remarks, Applicant respectfully requests reconsideration and allowance of the subject application.

§112 Rejections

Claims 27-31 are rejected under 35 U.S.C. §112, first paragraph, as allegedly failing to comply with the written description requirement. The Office asserts that the claims contain subject matter which was not described in the specification in such a way as to reasonably convey to one skilled in the relevant art that the inventor(s), at the time the application was filed, had possession of the claimed invention. Specifically, the Office asserts that the term "interpreting the command based on the current operation environment of the command line interface" in claim 27 was not described in the specification. Applicant respectfully traverses the rejection.

Burden on the Office

The Office has failed to establish a prima facie case of lack of enablement. More particularly, the Office has failed to satisfy its foundational burden for a "lack of enablement" rejection. Therefore, the Office must withdraw the rejections made under 35 U.S.C. §112, first paragraph.

According to MPEP §2164.04, the Office has the initial *burden* to establish a reasonable basis to question enablement in order to make a "lack of enablement" rejection. Applicant specifically notes that the specification is presumed to be enabling. MPEP §2164.04 states that "A specification disclosure which contains a teaching of the manner and process of making and using an invention in terms which correspond in scope to those used in describing and defining the subject matter sought to be patented *must* be taken as being in compliance with the enablement requirement of 35 U.S.C. 112, first paragraph, unless there is a reason to doubt the objective truth of the statements contained therein which must be relied on for enabling support" (emphasis added). Applicant submits that the Office has failed to provide a reason to doubt the presumption that the specification is enabling.

Furthermore, MPEP §2164.04 indicates that the language of the explanation provided by the Office should focus on those factors, reasons, and evidence that lead the examiner to conclude that the specification fails to teach how to make and use the claimed invention without undue experimentation, or that the scope of any enablement provided to one skilled in the art is not commensurate with the scope of protection sought by the claims. This can be done by making specific findings of fact, supported by the evidence, and then drawing conclusions based on these findings of fact. If the examiner believes that information is missing about one or more essential parts or relationships between parts which one skilled in the art could not develop without undue experimentation, according to MPEP §2164.04 "the examiner should *specifically identify* what information is missing and *why one skilled in the art could not supply the information without undue experimentation*" (emphasis added). Furthermore, according to MPEP §2164.04, "specific technical reasons [for the rejection] are *always* required" (emphasis added).

The Office has not met the initial *burden* of establishing a reasonable basis to question enablement. In addition, the Office has failed to give "specific technical reasons" to support a "lack of enablement" rejection. Instead, the Office simply asserts (Office Action, page 2, par. 2.a.) that the term "interpreting the command based on the current operation environment of the command line interface" in claim 27 was not described in the specification.

Although it is clear on the face of the assertions and statements provided by the Office that the Office has failed to meet its burden of establishing a reasonable basis to question enablement, Applicant nonetheless briefly discusses herein below how the disclosure, as filed, enables the claimed invention for one skilled in the art.

However, Applicant specifically notes that because the Office did not meet the fundamental requirements for its 35 U.S.C. §112, first paragraph rejection, Applicant has not been given a fair opportunity to respond to the rejection and it would therefore be inappropriate for the Office to make the next action final. More specifically, according to MPEP §2164.04 regarding the principles of compact prosecution, if an enablement rejection is appropriate, the first Office action on the merits should present the best case with all the relevant reasons, issues, and evidence so that all such rejections can be withdrawn if applicant provides appropriate convincing arguments and/or evidence in rebuttal. Providing the best case in the first Office action will also allow the second Office action to be made final should applicant fail to provide appropriate convincing arguments and/or evidence. Citing new references and/or expanding arguments in a second Office action could prevent that Office action from being made final. The principles of compact prosecution also dictate that if an enablement rejection is appropriate and the examiner recognizes limitations that would render the claims enabled, the examiner should note such limitations to applicant as early in the prosecution as possible.

The Disclosure, As Filed, Enables The Claimed Invention

Applicant's **claim 27** recites

> A method comprising:
>
> receiving a command through a command line interface;
>
> fetching an alias for the command;
>
> interpreting the command based on the alias and the current operating environment of the command line interface;
>
> executing the command as one or more WMI API calls against a target namespace;
>
> receiving WMI data in XML form;
>
> applying an XSL style sheet format the WMI data; and
>
> presenting the WMI data through the command line interface.

The Office asserts that the term "interpreting the command based on the current operation environment of the command line interface" was not described in the specification. First of all, Applicant's claim recites "interpreting the command based on *the alias and* the current operat*ing* environment of the command line interface" (emphasis added), and any description of this phrase should be viewed in its proper context, which includes the entire phrase.

Secondly, Applicant's specification clearly describes the claimed subject matter in such a way as to reasonably convey to one skilled in the relevant art that the inventor(s), at the time the application was filed, had possession of the claimed invention. For example, Applicant's specification recites the following at page 17, line 5 - page 18, line 6:

> The WMI schema exposed by the WMI infrastructure **400** is made visible to the user of the WMI command line utility at a management station **202** through an intermediate alias object. An alias object is effectively a command that is executed on the command line utility in order to capture the features of a target WMI class and to facilitate a specific administrative task, such as managing a system

> process, configuring a netcard, or discovering CPU utilization. Alias objects are instances of well-defined, command-related classes that are organized into a command schema **404**, as illustrated in **Fig. 5**. The command schema **404** drives the WMI command line utility and defines the commands used in the utility. *That is, the command line utility uses the class definitions or aliases in the command schema 404 to interpret the command information entered by a user and apply that command interpretation against the target WMI schema.*
>
> *The WMI command line utility and its underlying command schema 404 also permit the organization of commands by roles, so that administrators needing to perform specific administrative tasks are able to focus on a specific set of commands without being faced with the complete set of commands that make up the entire command schema 404. The command schema 404 is logically located in a default namespace structure on the management station 202, although it can exist in any namespace on any machine and is not limited to the management station 202. The namespace structure provides a logical grouping of classes and class instances that is intended to reflect the organization of a company's operational environment. Since the operational environment can differ significantly from one company to the next, it is expected that the corresponding namespace structures will also vary substantially from one organization to another. Thus, a given organization is able to organize commands based on suitable administrative roles by creating logical namespace in which relevant alias objects or command class instances can be found.*
>
> (emphasis added).

Thus, the command line utility interprets command information based on aliases as well as operational environments that indicate how commands are organized according to administrative roles.

Based at least on the above passage from Applicant's specification, it is clear that claim 27 is supported by a specification that describes the claimed subject matter such that one skilled in the art to which it pertains is enabled to make and/or use the invention. The rejection to claim 27 and its dependent claims 28-31 based on 35 U.S.C. §112, first paragraph, should therefore be removed.

In addition, Applicant respectfully requests that any future rejections by the Office which are similarly based on 35 U.S.C. §112, first paragraph, satisfy the fundamental requirements for such rejections so that Applicant has a full and fair opportunity to respond specifically to such rejections.

§103 Rejections

Claims 1-26, 33 and **37** are rejected under 35 U.S.C. §103(a) as allegedly being unpatentable over US Patent No. 6,629,128 to Glass in view of US Patent No. 6,560,591 to Memmott et al. (hereinafter, "Memmott"). Applicant respectfully traverses the rejection.

Applicant's **claim 1** recites:

> A command line utility embodied in one or more computer-readable media, the command line utility comprising:
>
> an object model command schema to define a mapping between one or more commands and an object model target schema, the one or more commands generated by the command schema and configured to operate against the target schema through the command line utility.

Glass teaches a system for distributed processing in a computer network that includes, a client side object request broker executing on a client computer and a server-side object request broker executing on a server computer. The system provides communications between objects in different address spaces connected to a common network and generates remote proxies and other objects to provide communications across the network. The server computer is connected to the client computer through the network. A remote proxy generator generates remote proxy classes for client-side communications support for communications between a client application and a server object. The remote proxy generator resides in the server-side object request broker and instantiates the remote proxy class to create a remote proxy object. A client-side type generator generates a client side type object for a class of the server object. The client-side type object provides access to methods of the server object. A client-side function generator generates one or more client-side function objects for providing a connection to one or more methods of the server object. The one or more client-side function objects correspond in number to the one or more methods of the server object. A client-side reference generator generates a client-side reference object for encoding messages sent between the remote proxy object and the server object into a format of a communication protocol used by the server-side object request broker. The distributed processing system also includes a client-side streamer generator that generates a set of streamer objects corresponding in number to the methods of the server object. Each streamer object encodes a method invocation request for an associated server method into the format of the communicator protocol used by the server-side object request broker. (col. 3, ln. 66 - col. 4, ln. 49).

A server-side local reference generator generates a local reference object that includes an address of the server object and a type of the server object. A server-side type generator generates a server-side type object for the class of the server object. The server-side type address provides access to the methods of the server object. A server-side function generator generates one or more server-side function objects corresponding in number to the one or more client-side function objects. The one or more server-side function objects are linked to the server-side type object. (col. 4, lns. 29-38).

Regarding **claim 1**, the Office asserts, at page 3 of the Office Action, that "Glass teaches the invention substantially as claimed including: the command line utility (interface generator 250 is a command line predevelopment utility, col 19, ln 10-14/ Fig. 3/ 10/11), an object mode[l] command schema (client side type generator, col 17, ln 54-58/ col 18, ln 47-53), one or more commands (type object 170, col 17, ln 54-58/ function objects 210, col 18, ln 47-53), an object mode[l] target schema (the method of server object 110, col 17, ln 54-58, col 18, ln 47-53), an object mode[l] command schema to define correspondence between one ore [sic] more commands (col 17, ln 50-58/ col 18, ln 47-55), the one or more commands generated by the command schema and configured to operate against the target schema through the command line utility (col 17, ln 50-58/ col 18, ln 47-55)".

Among other things, however, Glass does not teach or suggest "an object model command schema" as recited in claim 1. The Office asserts that the "client side type generator" of Glass (col. 17, ln 54-58; col. 18, ln 47-53) teaches "an object model command schema" as recited in claim 1. However, the "client side type generator" of Glass is not a schema in any respect. A command schema includes a collection of classes which forms a template that is used to represent information about command aliases (Application specification, pg. 20, ln. 12-13). The command schema follows the industry standard CIM schema, which is a way to express management information that relies on inheritance and other object-oriented features for the reuse and standardization of object classes representing system devices. Schemas make significant use of inheritance to allow applications to treat groups of similar objects in the same way.

By contrast to the object model command schema of claim 27, the "client side type generator" of Glass is one of various object generation processes of a server-side object request broker (ORB) (114) (col. 17, ln. 15-17). An ORB is programming that acts as a "broker" between a client request for a service from a distributed object or component and the completion of that request. The ORB allows a client program to request a service without knowing where the server is in a distributed network or the exact nature of the interface to the server program. Thus, Glass does not teach or suggest "an object model command schema" as recited in claim 1, because the "client side type generator" of Glass has no relation at all to an object model schema, but is instead, an object generation process of an object request broker (ORB).

In addition, Glass does not teach or suggest "an object model target schema" as recited in claim 1. The Office asserts that the "method of server object 110" of Glass (col. 17, ln. 54-58, col. 18, ln. 47-53) teaches "an object model target schema" as recited in claim 1. However, a "method of server object 110" of Glass is not a schema in any respect. An object model target schema represents an enterprise through target objects in an object-oriented model that follows the industry standard CIM schema (Application specification, pg. 6, lns. 7-25). The CIM schema provides a way to express management information that relies on inheritance and other object-oriented features for the reuse and standardization of object classes representing system devices. Schemas make significant use of inheritance to allow applications to treat groups of similar objects in the same way.

By contrast, a method or methods of a "server object 110" in Glass, are merely procedures included in the server object. In general, methods provide instructions for manipulating an object based on relevant data in the object.

Methods are not schemas that represent target objects. The "method of server object 110" of Glass therefore does not teach "an object model target schema" as recited in claim 1. The Office points to nothing in either of the cited reference that teaches or suggests "an object model command schema" or "an object model target schema" as recited in claim 1.

Furthermore, there is no teaching or suggestion in Glass of "one or more commands generated by the command schema" or that such commands are "configured to operate against the target schema through the command line utility". The Office again points to Glass at col. 17, ln. 50-58, and col. 18, ln. 47-55, to support its assertion that Glass teaches "one or more commands generated by the command schema and configured to operate against the target schema through the command line utility". However, there simply is no such teaching found here or anywhere else in Glass. Glass states the following at col. 17, ln. 50-58:

> Interface generator 250 and remote enabling classes without interfaces are discussed in the following section.
>
> Client-side type generator 302 generates type object 170 using class information obtained from server object 110. Type object 170 represents the class of server object 110 and includes an array of function objects 172 that provide access to the methods of server object 110.

At col. 18, ln. 47-55, Glass states the following:

> Server-side function generator 314 generates function objects 210 or specialized function objects such as EJBfunction objects 206. Function objects 210 or EJB function objects 206 correspond in number to the methods of server object 110. Each function object 210 or EJB function object 206 directly invokes a corresponding method on server object 110. Each EJBfunction object 206 is instantiated from a standard EJBfunction class that provides common

> functionality in addition to the functionality of function object 210. Unique functionality may be added to each EJBfunction object 206 after it has been instantiated to provide for unique processing needs included in function object 210.

The Office provides nothing to suggest that there is any relationship at all between these cited passages in Glass and Applicant's claim 1, which recites "one or more commands generated by the command schema and configured to operate against the target schema through the command line utility". If this rejection is to be maintained, Applicant respectfully requests some explanation as to any such relationship in order that Applicant is afforded a full and fair opportunity to respond to this rejection. Furthermore, if the next Action maintains such rejection, Applicant additionally requests that such Action not be made final so that Applicant may have a full and fair opportunity to respond to the rejection.

Continuing with respect to claim 1, the Office admits at page 3 of the Office Action, that Glass does not teach "a correspondence as a mapping" (i.e., the "object model command schema to define a mapping" and the "mapping between one or more commands and an object model target schema" as recited in claim 1). Instead, the Office relies on Memmott for such teaching. The Office asserts that Memmott's "mapping of at least a portion of the query received in task into the namespace of the data provider indicated by the corresponding data provider identifier" at col. 5, ln. 50-55 and col. 9, ln. 64-68, teaches Applicant's claimed "object model command schema to define a mapping" and that such mapping is "mapping between one or more commands and an object model target schema".

Memmott teaches a system for managing data providers. A data requester 110 forwards a query to a data resolver 120, which chooses a priority list of data providers 130 from a set of lists based on the characteristic of the query. The data resolver 120 forwards a request to a data provider 130 in the list based on the query. The data resolver 120 receives data in response to the request and returns a response to the data requestor 110 based on the data. (col. 3, ln 7-62; col. 4, ln 1-21).

Regarding Applicant's claim 1, the Office has not pointed to anything in Memmott or any other reference that teaches or suggests an "object model command schema to define a mapping" or that such mapping is a "mapping between one or more commands and an object model target schema". The Office asserts that Memmott teaches "a mapping" at col. 5, ln. 50-55 and col. 9, ln. 64-68. However, as noted, Applicant's claim 1 recites "a mapping between one or more commands and an object model target schema". By contrast, Memmott teaches a string that represents a mapping between a portion of a query and a data provider identifier. This is not the same as "a mapping between one or more commands and an object model target schema". Furthermore, Memmott does not teach or suggest "an object model command schema to define a mapping" as recited in Applicant's claim 1. Like Glass, Memmott does not teach "an object model command schema" at all. Thus, it cannot fairly be said that Memmott teaches "an object model command schema to define a mapping". It further cannot fairly be said that Memmott teaches "an object model command schema to define a mapping between one or more commands and an object model target schema", as recited in Applicant's claim 1.

A prima facie case of obviousness requires that the prior art reference (or references when combined) must teach or suggest all the claim limitations (MPEP 2142, 2143). However, it is clear from the above discussion, that various elements recited in Applicant's claim 1 are not taught or suggested by Glass and Memmott, alone or in combination. Furthermore, the various elements discussed above and recited in Applicant's claim 1 are not taught or suggested by any other references relied upon by the Office. For at least the numerous reasons above showing that Glass and Memmott, alone or in combination, fail to teach or suggest all the claim limitations of claim 1, a prima facie case of obviousness is not supported. Applicant therefore respectfully requests that the §103(a) rejection to claim 1 be removed.

Claims 2-15 depend from claim 1 and therefore include the elements of claim 1. Therefore, claims 2-15 are allowable at least on the basis of this dependency, in addition to the further elements recited therein which are neither shown nor suggested by the cited references. Accordingly, Applicant respectfully requests that the 35 U.S.C. §103(a) rejection to claims 2-15 be removed.

Independent **claim 16** recites:

> An object model schema embodied in one or more computer-readable media, the object model schema comprising:
>
> an alias class to define alias instances, each alias instance representing a command;
>
> a verb class to define verb instances, each verb instance representing behavior available through an alias instance;
>
> a parameter class to define parameters accepted by a verb instance;
>
> a format class to define format instances, each format instance having a list of properties to be displayed through an alias instance;
>
> a property class to define property instances, each property instance representing a property value from a property list;
>
> a connection class to define connection instances, each connection instance representing connection parameters used by an alias instance to establish a connection to the target schema;
>
> a qualifier class to define qualifier instances, each qualifier instance representing constraints on elements of an alias instance;
>
> a localized string class to define localized string

> instances, each localized string instance representing a text localization for translating text into a localized language; and
>
> a see-also association to associate an alias instance with other related alias instances.

On page 5 of the current Office Action, the Office rejects independent **claim 16** for the same reasons it rejects claim 12. Furthermore, the Office rejects claim 12 for the same reasons it rejects claims 2-8. Regarding claim 2, the Office asserts that Memmott teaches an alias class at col. 5, ln 18-30 and col. 4, ln 40-60. However, Memmott merely discusses query characteristics that indicate different classes, e.g., class 1 and class 2. Nowhere does Memmott teach or suggest "an alias class to define alias instances, each alias instance representing a command" as recited in Applicant's claim 16. Thus, the rejection of claim 16 is not supported and should be removed.

Further regarding the rejection of claim 16, the Office asserts that with respect to claim 3, Memmott teaches a verb class, a format class, and a connection class as a subclass at col. 5, ln 17-30 and col. 4 ln 40-60. However, the words "verb class", "format class", and "connection class" do not appear in any form throughout the entire text of Memmott. Furthermore, there is no discussion whatsoever in Memmott that teaches, suggests, or implies a "verb class", "format class", or a "connection class". As shown above, Applicant's claim 16 recites,

> a verb class to define verb instances, each verb instance representing behavior available through an alias instance;
>
> a format class to define format instances, each format instance having a list of properties to be displayed through an alias instance;
>
> a connection class to define connection instances, each connection instance representing connection parameters used by an alias instance to establish a connection to the target schema;

For the additional reasons that Memmott does not teach or suggest a "verb class to define verb instances, each . . . ", "format class to define format instances, each . . . ", or a "connection class to define connection instances, each . . . " as recited in Applicant's claim 16, the rejection of claim 16 is not supported and should be removed.

Further regarding the rejection of claim 16, the Office asserts that with respect to claim 4, Memmott teaches at col. 5, ln 18-30, a parameter class as a subclass with each instance of the parameter class representing parameters. However, claim 16 recites,

> a verb class to define verb instances, each verb instance representing behavior available through an alias instance;
>
> a parameter class to define parameters accepted by a verb instance;

As noted above, Memmott does not teach or suggest "a verb class to define verb instances". Thus, it cannot fairly be said that Memmott teaches "a parameter class to define parameters accepted by a verb instance". Furthermore, the words "parameter class" do not appear in any form throughout the entire text of Memmott. Moreover, there is no discussion in Memmott that teaches, suggests, or implies a "parameter class to define parameters accepted by a verb instance". Accordingly, for the additional reason that Memmott does not teach or suggest a "parameter class" as recited in Applicant's claim 16, the rejection of claim 16 is not supported and should be removed.

Further regarding the rejection of claim 16, the Office asserts that with respect to claim 5, Memmott teaches at col. 5, ln 18-30, a property class as a subclass to the format class with each instance of the property class representing a property value. With respect to a property class, claim 16 recites,

> a format class to define format instances, each format instance having a list of properties to be displayed through an alias instance;
> a property class to define property instances, each property instance representing a property value from a property list;

First of all, there is no discussion or teaching in Memmott regarding a "format class to define format instances, each format instance having a list of properties to be displayed through an alias instance". Thus, it cannot fairly be said that Memmott teaches "a property class to define property instances, each property instance representing a property value from a property list". Furthermore, the words "property class", "property instances", "property list", "format class", etc., do not appear in any form throughout the entire text of Memmott. Moreover, there is no discussion whatsoever in Memmott that teaches, suggests, or implies anything about a "property class". Accordingly, for these additional reasons, the rejection of claim 16 is not supported and should be removed.

The very same arguments stated above regarding certain elements of claim 16, can be equally applied to the various other elements of claim 16. That is, Memmott does not teach, suggest, or imply anything regarding elements including "a qualifier class", "a localized string class", or "a see-also association".

For at least all the numerous reasons stated above, the rejection of claim 16 is not supported. Accordingly, Applicant respectfully requests that the rejection to claim 16 be removed.

Regarding independent **claim 17**, the Office rejects claim 17 for the same reasons it rejects claim 1. The elements of claim 17 parallel those discussed above with respect to claim 1. For example, claim 17 recites in part:

> a set of commands generated by an object model command schema to operate against an object model target schema, the command schema defining a mapping between the set of commands and the target schema; and
>
> an interface utility to facilitate implementation of individual commands within the set of commands.

Therefore, the reasoning stated herein above regarding the rejection of claim 1 is similarly applicable to the rejection of claim 17. For example, none of the cited references teaches or suggests "a set of commands generated by an object model command schema". Accordingly, for at least the various reasons stated above regarding claim 1, Applicant respectfully submits that a prima facie case of obviousness is not supported with respect to claim 17. Applicant therefore respectfully requests that the §103(a) rejection to claim 17 be removed.

Claims 18-23 depend from claim 17 and therefore include the elements of claim 17. Therefore, claims 18-23 are allowable at least on the basis of this dependency, in addition to the further elements recited therein which are neither shown nor suggested by the cited references. Accordingly, Applicant respectfully requests that the 35 U.S.C. §103(a) rejection to claims 18-23 be removed.

Regarding independent **claim 24**, the Office rejects claim 24 for the same reasons it rejects claims 1 and 9-12. Claim 24 recites the following:

> A management application embodied in one or more computer-readable media, the management application comprising:
>
> a first object model to control the configuration and behavior of the management application in operating against and managing a second object model.

Although the Office rejects claim 24 for the same reasons it rejects claims 1 and 9-12, in its rejection of claim 1 and 9-12, the Office does not point out anything in any of the cited references that teach or suggest the elements of claim 24. Furthermore, a thorough review of the cited references reveals that the references do not teach or suggest the elements of claim 24. Specifically, none of the cited references teaches or suggests at least "a first object model to control the configuration and behavior of the management application in operating against and managing a second object model" as recited in claim 24. Accordingly, the rejection of claim 24 cannot stand, and Applicant respectfully requests that the 35 U.S.C. §103(a) rejection to claim 24 be removed.

Claims 25-26 depend from claim 24 and therefore include the elements of claim 24. Therefore, claims 25-26 are allowable at least on the basis of this dependency, in addition to the further elements recited therein which are neither shown nor suggested by the cited references. Accordingly, Applicant respectfully requests that the 35 U.S.C. §103(a) rejection to claims 25-26 be removed.

The Office also rejects **claim 33** based on Glass and Memmott. Regarding independent claim 33, the Office rejects claim 33 for the same reasons it rejects claim 1. Claim 33 recites the following:

> A method of managing objects in a target schema comprising:
> providing a user interface;
> defining a command structure through an object-oriented command schema, the command schema including an alias class;
> instantiating an object of the alias class as an alias by receiving parameters of the alias class through the user interface, the alias representing a command which maps to an object in the target schema; and
> executing the command against the object in the target schema.

To the extent elements of claim 33 parallel elements recited in claim 1 (e.g., "defining a command structure through an object-oriented command schema, the command schema including an alias class"), arguments already presented above regarding the rejection of claim 1 apply similarly to the rejection of claim 33. For these reasons alone, claim 33 is allowable over Glass and Memmott and the rejection to claim 33 should be removed.

Furthermore, there is no teaching or suggestion in Glass and Memmott or any other cited reference of the various additional elements of claim 33, including, "instantiating an object of the alias class as an alias by receiving parameters of the alias class through the user interface", or "the alias representing a command which maps to an object in the target schema". The Office Action does not point to anything in the cited references that teaches or suggests these elements of claim 33. Further, Applicant is unable to find any teaching or suggestion of such elements in any of the cited references. Accordingly, for these additional reasons, claim 33 is allowable over the cited references and the rejection to claim 33 should be removed.

The Office also rejects **claim 37** based on Glass and Memmott. Regarding independent claim 37, the Office rejects claim 37 for the same reasons it rejects claim 33. Accordingly, the same arguments regarding claim 33 from above apply equally to claim 37. Thus, for at least these same reasons, claim 37 is allowable over the cited references and the rejection to claim 37 should be removed.

The Office rejects **claims 27-32** under 35 U.S.C. §103(a) as allegedly being unpatentable over Memmott in view of Glass and further in view of Steve (Network and System Management with XML) (hereinafter, "Steve"). Applicant respectfully traverses the rejection.

Claim 27 recites, in part, the following:

> receiving a command through a command line interface;
> fetching an alias for the command;
> interpreting the command based on the alias and the current operating environment of the command line interface;
> executing the command as one or more WMI API calls against a target namespace;
> receiving WMI data in XML form;
> applying an XSL style sheet format the WMI data; and
> presenting the WMI data through the command line interface.

Regarding **claim 27**, the Office asserts that Memmott teaches all the elements of claim 27 except the "WMI API". The Office points to Memmott at various locations in cols. 3, 4, 5, 6, 8, and 9. However, Memmott does not teach or suggest "fetching an alias for the command", where the "command" is received "through a command line interface", as generally recited in claim 27. The Office does not point to anything in Memmott or any other reference that teaches or suggests such elements. In addition, Memmott does not teach or suggest "interpreting the command based on the alias and the current operating environment of the command line interface", where the "command" is received "through a command line interface" and the "alias" is fetched "for the command". In addition, Memmott does not teach or suggest "executing the command as one or more WMI API calls against a target namespace", where the "command" is received "through a command line interface". In addition, Memmott does not teach or suggest "receiving WMI data in XML form", or "applying an XSL style sheet format the WMI data", or "presenting the WMI data through the command line interface", all as recited in Applicant's claim 27. The Office has not pointed to anything in Memmott or any other reference that teaches or suggests these elements as recited in claim 27.

The Office is invited to point to ***specific locations*** in Memmott or within any of the other cited references, where such elements of claim 27 are taught, suggested, or implied in any way. Applicant respectfully submits that such teachings, suggestions, or implications, do not exist in any of the cited references. Accordingly, the rejection of claim 27 cannot stand, and Applicant respectfully requests that the §103(a) rejection of claim 27 be removed.

Furthermore, regarding claim 27, the Office admits that Memmott and Glass do not teach an XSL style sheet, and refers to Steve (pg. 4 of 8, ln. 38-45 to page 5 of 8, ln. 1-8) for support of such teaching. The Office asserts that Steve teaches a command line and an XSL style sheet. Steve provides a very broad statement regarding "Network and Systems Management with XML". Included in Steve are general discussions of CIM and XML. Steve mentions on pg 4, ln 38 - pg 5, ln 8, that "A forthcoming new standard . . . is the Extensible Style Language (XSL)" and that "a command-line interface could be displayed as a style-sheet-defined view . . . expressed with XML". However, claim 27 recites "fetching an alias for the command", where the "command" is received "through a command line interface". Such elements are not taught or suggested by Steve, and as noted above, are also not taught or suggested by the other cited references. Claim 27 further recites, "interpreting the command based on the alias and the current operating environment of the command line interface", where the "command" is received "through a command line interface" and the "alias" is fetched "for the command". Claim 27 also recites, "executing the command as one or more WMI

API calls against a target namespace", where the "command" is received "through a command line interface". Steve does not teach or suggest these elements.

For these additional reasons, Applicant respectfully submits that a prima facie case of obviousness is not supported with regard to Applicant's claim 27, and respectfully requests that the §103(a) rejection to claim 27 be removed.

Claims 28-32 depend from claim 27 and therefore include the elements of claim 27. Therefore, claims 28-32 are allowable at least on the basis of this dependency, in addition to the further elements recited therein which are neither shown nor suggested by the cited references. Accordingly, Applicant respectfully requests that the 35 U.S.C. §103(a) rejection to claims 28-32 be removed.

The Office also rejects **claims 34-36** under 35 U.S.C. §103(a) as allegedly being unpatentable over Glass in view of Memmott and further in view of Steve. Applicant respectfully traverses the rejection.

As noted above regarding claim 33, Glass and Memmott fail to teach or suggest the elements of claim 33. Applicant further notes that Steve does not remedy the deficiencies of Glass and Memmott and that claim 33 is allowable over the combination of these 3 references.

Claims 34-36 depend from claim 33 and therefore include the elements of claim 33. Therefore, claims 34-36 are allowable at least on the basis of this dependency, in addition to the further elements recited therein which are neither shown nor suggested by the cited references. Accordingly, Applicant respectfully requests that the 35 U.S.C. §103(a) rejection to claims 34-36 be removed.

Conclusion

All pending claims are in condition for allowance. Applicant respectfully requests reconsideration and prompt issuance of the subject application. If any issues remain that prevent issuance of this application, the Examiner is urged to contact the undersigned attorney before issuing a subsequent Action.

Respectfully Submitted,

Dated: 2/25/05 By: Nathan R. Rieth

Nathan R. Rieth
Reg. No. 44302
(509) 324-9256; X233